MASSIVE PARALLELISM AT NAS

Horst D. Simon

Computer Sciences Corporation

NAS Applied Research Branch

NASA Ames Research Center

Moffett Field, CA 94035

simon@orville.nas.nasa.gov

Supercomputing USA/Pacific '91

Santa Clara, California

June 19 - 21, 1991

Long Range Goals of the Numerical Aerodynamic Simulation (NAS) Program at NASA Ames

- Pursue an aggressive program in acquisition and testing of highly parallel computer systems.
- Explore architecture, algorithm, performance and language issues, especially for CFD applications.
- Bring the power of highly parallel supercomputers into the mainstream of scientific computing.
- Achieve one TFLOPS sustained performance on significant aerophysics applications by the year 2000.

NAS PROGRAM GOALS FOR PARALLEL SYSTEMS (1988)

Year of Installation	1989	1991	1994	1997
System[2]	Gen 1	Gen 2	Gen 3	Gen 4
Sustained Computing Rate (GFLOPS)	1	10	200	2000
Peak[1] Computing Rate (GFLOPS)	10	100	2000	20000
Main memory (Gbytes)	0.5	8	200	1000

[1]) Maximum rate that is assumed necessary to reach stated sustained rate

[2]) Computing rates and memory capacities are for full-scale systems

Current NAS Parallel Supercomputer Resources

System Name	Number FP Proc.	Memory (MB)	Peak MFLOPS	Last Upgrade
Cray-2	4	2048	2000	1988
Cray Y-MP	8	1024	2666	1989
TMC CM-2	1024	4096	14000	1991
Intel iPSC/860	128	1024	7680	1990

What have we learned from our experience with these systems?

Overview of Parallel CFD Research at NASA Ames

Project	Researchers	Y-MP	CM-2	iPSC
Multigrid (NAS b'mark)	Frederickson, Barszcz	x	x	x
Conj. gradient (NAS b'mark)	Schreiber, Simon	x	x	x
3D FFT PDE (NAS b'mark)	Bailey, Frederickson	x	x	x
Integer sort (NAS b'mark)	Dagum	x	x	x
LU solver (NAS b'mark)	Fatoohi, Venkatakrishnan	x	x	x
Scalar penta. (NAS b'mark)	Barszcz, Weeratunga	x	x	x
Block tridiag. (NAS b'mark)	Barszcz, Weeratunga	x	x	x
INS3D (incom. Nav. Stokes)	Fatoohi, Yoon	x	x	x
Isotropic turbulence simul.	Wray, Rogallo	x	x	x
PSIM3 (particle method)	McDonald	x		x
PSICM (particle method)	Dagum		x	
F3D (ARC3D mult. zones)	Barszcz, Chawala, Weeratunga		x	x
CM3D (ARC3D derivative)	Levit, Jesperson	x	x	
ARC2D	Weeratunga			x
Unstructured Euler solver	Hammond, Barth, Venkatakrishnan		x	x
Unstructured partitioning	Simon			x
High precision vortex anal.	Bailey, Krasny			x

The NAS Connection Machine-2

- Hardware has been upgraded with 64-bit floating point processors and 4 GB of main memory.
- Software has been upgraded to support the "slicewise" model.
- Most programmers now use the CM slicewise Fortran compiler, which is based on Fortran-90.

Notable application result:

- CM3D, a 3-D fully implicit Navier-Stokes CFD application. Performance: up to 275 MFLOPS (16K processors). C. Levit and D. Jesperson of NASA Ames.

Principal Advantages of the CM-2

- A programming language based on Fortran-90.
- Relatively stable system software.

Principal Disadvantages of the CM-2

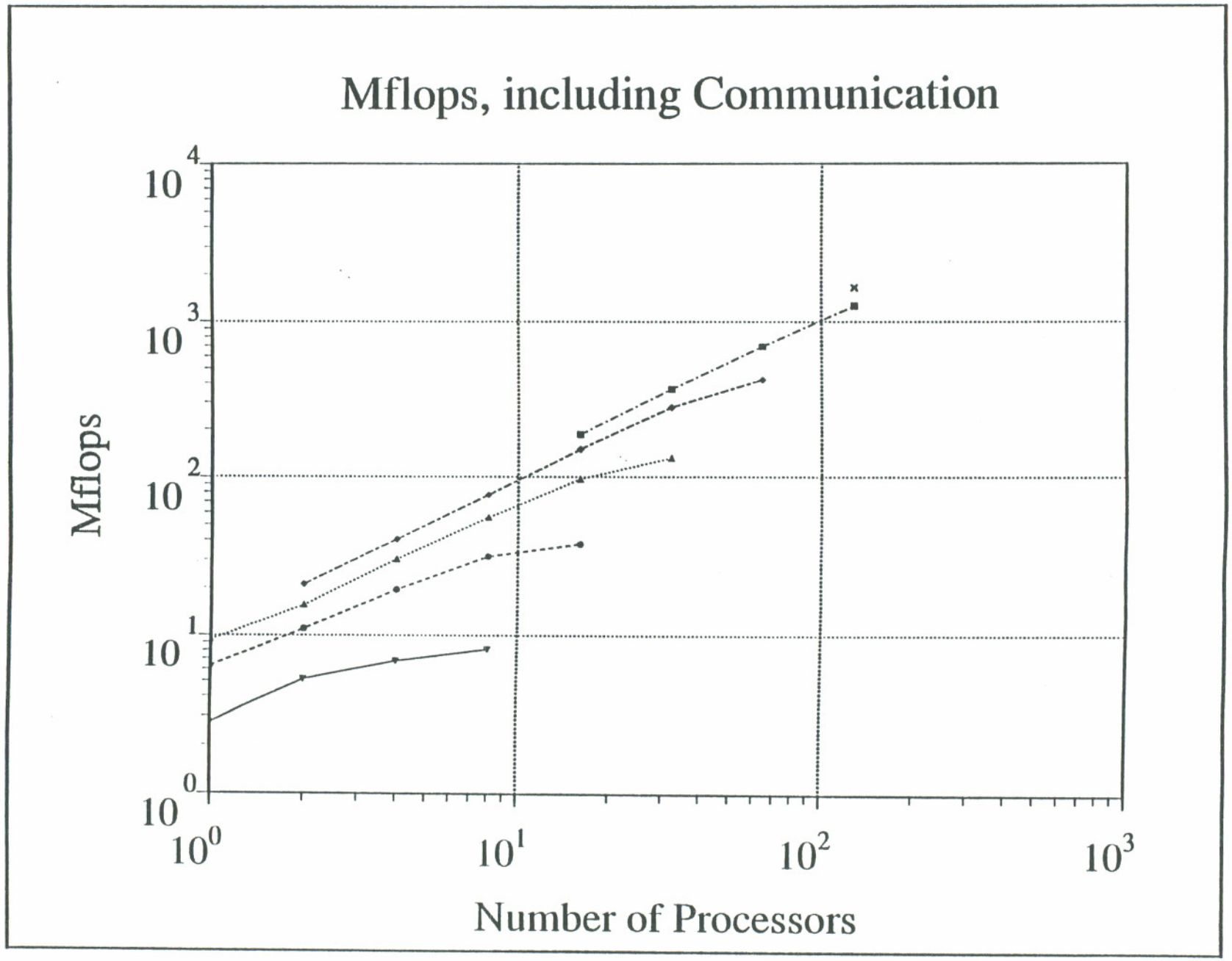
- Insufficient bandwidth between processors and local main memory.
- Insufficient bandwidth between nodes.
- Inflexible facility for partitioning nodes between users.
- Numerous bugs in Fortran compiler.
- Inefficient implementation of many Fortran-90 constructs.
- Poorly documented "tricks" are usually necessary to obtain respectable performance.

The NAS Intel iPSC/860

- Based on the Intel i860 RISC floating point processor, which features a peak performance of 60 MFLOPS (64-bit).
- 128 nodes with 8 MB main memory per node.
- Fortran and C compilers from the Portland Group can run on Sun or SGI workstations.

Notable application result:

- An isotropic turbulence simulation code. Performance: up to 1.6 GFLOPS (32-bit data, Vectoral language with assembly-coded 1-D FFT). A. Wray and B. Rogallo of NASA Ames.


Mflops, including Communication
Mflops
10^4
10^3
10^2
10^1
10^0
10^0
10^1
10^2
10^3
Number of Processors

Principal Advantages of the iPSC/860

- Straightforward to obtain "respectable" performance (i.e. > 100 MFLOPS).
- Flexible design for partitioning the system between users.

Principal Disadvantages of the iPSC/860

- Unstable operating system.
- Disappointing single node performance.
- Insufficient bandwidth between processors and local main memory.
- Insufficient bandwidth between nodes.
- Users must manually synchronize operations, decompose arrays and communicate data.

Problems with Conventional Benchmarks for Parallel Computers

- Rigid tuning requirements.
- Lack of automatic tools for converting "dusty deck" Fortran codes for parallel computers.
- Inappropriately small problem sizes.
- Inappropriate algorithms and implementation techniques.

The NAS Parallel Benchmarks

- Each problem is completely specified in "pencil and paper" fashion in a technical document.
- Implementations must be based on Fortran-77 or C, but a wide range of parallel constructs are allowed.
- With a few exceptions, assembly code and assembly language routines are prohibited for computations.
- Algorithms, implementation techniques and language constructs may be selected for a particular system.
- A set of single processor Fortran-77 codes is available as a starting point.

Available by sending e-mail to par-comp@nas.nasa.gov.

Brief Description of the NAS Parallel Benchmarks

1. An "embarrassingly parallel" Monte Carlo simulation problem.
2. A simplified multigrid computation.
3. A conjugate gradient eigenvalue computation involving unstructured matrices.
4. A 3-D partial differential equation solution using FFTs.
5. A large integer sort, used in "particle method" codes.
6. A block lower and upper triangular system solver.
7. A multiple scalar pentadiagonal equation solver.
8. A multiple block tridiagonal equation solver.

The last three are "simulated CFD application" benchmarks.

Preliminary Performance Results (MFLOPS)

Benchmark	Problem Size	Y-MP 8	CM-2 32K	iPSC/860 128
Embarrassingly Parallel	2^{28}	1104	436	362
Conjugate Gradient	2×10^6	154	2	70
3-D FFT PDE	$256^2 \times 128$	1459	273	498
LU solver	64^3	1365	151	*123
Scalar penta. solver	64^3	1356	39	*146
Block tridiagonal solver	64^3	1402	119	*200

MFLOPS figures are based on single processor implementations.

* These results are for 64 nodes.

3-D FFT PDE Benchmark Techniques

For each system, the key to implementing this benchmark was devising an appropriate technique for 3-D FFTs:

- Cray Y-MP: Declare array with physical dimensions $(n_1 + 1, n_2 + 1, n_3 + 1)$ and then perform vectorized FFTs in each dimension.
- iPSC/860: Perform 1-D FFTs along first dimension, transpose second dimension to first, 1-D FFTs, transpose, 1-D FFTs, transpose.
- CM-2: Employ the slicewise library routine FFT3D.

3-D FFT PDE Benchmark Performance Rates

System	Code	No. Proc.	Time (sec.)	MFLOPS	Speedup
Y-MP	F	1	39.23	192.2	1.00
	F	8	5.17	1458.6	7.59
	L	1	29.31	257.3	1.00
	L	8	6.15	1226.2	4.77
Intel	F	128	22.22	339.4	
	L	128	15.13	498.5	
CM-2	LE	16K	110.88	68.0	
	LB	16K	53.41	141.2	1.00
	LE	32K	87.87	85.8	
	LB	32K	27.63	272.9	1.93

F: All-Fortran code.
L: Uses a library FFT routine.
B: Busy times
E: Elapsed times

Sustained Performance Per Dollar

System	Peak MFLOPS	FFT PDE MFLOPS	Ratio (%)	Price (US $)	PDE MFLOPS Per million $
Y-MP/8	2666	1459	54.7	25.0	58
iPSC/860	7680	339	6.7	2.3	147
CM-2	14000	273	2.0	5.0	54

Reasons for low sustained-to-peak percentages:

- Immature compilers.
- Insufficient bandwidth between floating point processors and local main memory.
- Insufficient interprocessor network bandwidth.

Algorithms: MFLOPS vs. Run Time

NCUBE-2 Performance on a Convection-Diffusion Problem
(Shadid and Tuminaro, Sandia Natl. Lab.)

Solver Algorithm	Floating Point Operations	CPU Time (Secs.)	MFLOPS
Jacobi	3.82×10^{12}	2124	1800
Gauss-Seidel	1.21×10^{12}	885	1365
Least Squares	2.59×10^{11}	185	1400
Multigrid	2.13×10^{09}	6.7	318

Conclusions:

- When selecting an algorithm for a parallel computer, fundamental numerical efficiency is much more important than appropriateness for a particular architecture.
- Parallel computer systems must be designed to run numerically efficient algorithms at respectable performance rates.

Hardware: SIMD vs. MIMD

- The data parallel model has proven suitable for many NASA applications. The trivial synchronization of SIMD hardware has an advantage for these.

- Possible exceptions: complicated geometries; chimera schemes; domain decomposition schemes. MIMD hardware may have an advantage for these.

- MIMD hardware may have a significant advantage in serving multiple users.

Conclusion: Neither purely SIMD nor purely MIMD hardware systems are ideal — an amalgam of the two would be best.

Language Obstacles Limiting Widespread Usage of Parallel Computers

- Time required to port, debug and tune codes.
- Fear that ported codes will run on only one system.
- Need for explicitly programming data communication.
- Need for explicitly decomposing data arrays.
- Need to master obscure details of the architecture and arbitrary tuning "tricks".
- Difficulties in debugging asynchronous computations.

Fortran-90: a Data Parallel Programming Language

- Most data parallel operations can be represented using Fortran-90 array constructs.
- Array constructs eliminate the need for explicit array decomposition.
- Many important communication operations, including array transpositions and shift operations, are provided.
- Standardization is needed for data array layout directives, etc.
- For true asynchronous computations, no language paradigm has yet emerged. Additional research is needed.

How Much Parallelism Can We Effectively Use?

- Many 2-D applications exhibit only 1-D parallelism.
- Many 3-D applications exhibit only 2-D parallelism.
- Many 2-D and 3-D applications require as many as 50 data words per grid element.

Conclusion: n words of memory can support at most $(n/50)^{1/2}$-way or $(n/50)^{2/3}$-way parallelism.

Systems which require more than this level will have limited usability.

A Call for Higher Standards in Reporting Performance

Many scientific papers and presentations distort performance figures:

- 32-bit results are compared with 64-bit results on other systems.
- MFLOPS figures are based on operation counts of inflated parallel implementations and inefficient algorithms.
- Codes used for comparative performance on conventional supercomputers have not been fully tuned.
- Inner kernel performance figures are quoted for an entire application.
- Assembly code, etc., is employed but not disclosed.
- Figures obtained on smaller systems are projected linearly to full-sized systems, without justification.

Reference: "Twelve Ways to Fool the Masses When Giving Performance Results on Parallel Computers", by DHB.

Conclusions

- The CM-2 and iPSC/860 systems, while showing promise, do not yet deliver sustained performance comparable to full Cray systems.
- Common shortcomings: immature compilers and operating systems; insufficient main memory bandwidth; insufficient internode bandwidth.
- Parallel computer systems must be designed to run numerically efficient algorithms at respectable performance rates.
- The best parallel hardware design is an amalgam between SIMD and MIMD.
- Fortran-90 is good for data parallel computations. No paradigm has yet emerged for true asynchronous computations.
- There is a limit to the amount of parallelism present in many large scientific computations.
- The field of parallel computing may lose credibility unless researchers and vendors are more circumspect in reporting performance.